

c0a2311499 / ProjExD\_Group14

🔍

📁

👤

<> Code

🔔 Issues

🔗 Pull requests

🔄 Actions

📁 Projects

📖 Wiki

🛡 Security

📈 Insights

⚙

ProjExD\_Group14 / README.md

...

c0a2311499

C0A23114

追加機能実装完了

6248412 · now

💬

🕒

29 lines (23 loc) · 2.21 KB

PreviewCodeBlame

Raw📄📥✎⌵☰

# こうかといん伝説（仮）

---

## 実行環境の必要条件

---

- python >= 3.10
- pygame >= 2.1

## ゲームの概要

---

- こうかといんを十字キーで操作しスペースキーでビームを発射、敵やその爆弾を打ち落としてスコアを稼いでいくゲームで、元あるこうかといん無双に機能を追加していくことでさらにゲーム感が増している。具体的にはこうかといんの体力や攻撃力、技を使うためのスキルポイントの追加、一定のスキルを取ることで挑戦可能なボス戦の追加、画面スクロールの追加をすることでゲーム感を増させている。このゲームはクリアを目指すというより、ハイスコアを目指すものである。

## ゲームの実装

---

### 共通基本機能

- 背景画像、こうかといん、こうかといんのビーム、敵のUFO、UFOの爆弾の描画
- こうかといんの各種スキルの実装
- スコアの描画

### 担当追加機能

- 画面の動き(担当：寺川 竣祐)：横スクロール、敵を右から表示させる機能の追加
- ライフゲージ(担当：川畑 しんのすけ)：こうかといんのライフゲージの追加
- ボス追加①(担当：濱口 莉奈)：通常画面からボス画面、ボス撃破後に通常画面への移動
- ボス追加②(担当：町田 拓斗)：ボスの性能を決めるクラスの作成
- ステータス変化(担当：萩原 颯人)：打ち落とした爆弾の色によって各種ステータスを変化

### ステータス変化（担当：萩原 颯人）

- 攻撃力の概念の追加：打ち落とした爆弾が赤色のときにこうかとの攻撃力を + 1 する。攻撃力の数値をを画面上に追加して可視化した。
- スキルポイントの概念の追加：打ち落とした爆弾が青色のときにスキルポイントが + 1 する。SP と書いて画面上に追加して可視化した。
- こうかтон無双ではスコアを消費して技を使っていたがスキルポイントを追加したので技に応じたポイントを消費して発動できるようにコードを書き直した。

## ToDo

- ☐ 体力の回復機能の追加

c0a2311499 /  
ProjExD\_Group14[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

ProjExD\_Group14 / musou\_kokaton.py



c0a2311499 C0A23114 追加機能実装完了

6248412 · now



500 lines (427 loc) · 17.8 KB

Code

Blame

Raw



```
1  import math
2  import os
3  import random
4  import sys
5  import time
6  import pygame as pg
7
8
9  WIDTH, HEIGHT = 1600, 900 # ゲームウィンドウの幅, 高さ
10 os.chdir(os.path.dirname(os.path.abspath(__file__)))
11
12
13  def check_bound(obj_rct:pg.Rect) -> tuple[bool, bool]:
14      """
15      Rectの画面内外判定用の関数
16      引数: こうかとんRect, または, 爆弾Rect, またはビームRect
17      戻り値: 横方向判定結果, 縦方向判定結果 (True: 画面内/False: 画面外)
18      """
19      yoko, tate = True, True
20      if obj_rct.left < 0 or WIDTH < obj_rct.right: # 横方向のはみ出し判定
21          yoko = False
22      if obj_rct.top < 0 or HEIGHT < obj_rct.bottom: # 縦方向のはみ出し判定
23          tate = False
24      return yoko, tate
25
26
27  def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
28      """
29      orgから見て, dstがどこにあるかを計算し, 方向ベクトルをタプルで返す
30      引数1 org: 爆弾SurfaceのRect
31      引数2 dst: こうかとんSurfaceのRect
32      戻り値: orgから見たdstの方向ベクトルを表すタプル
33      """
34      x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
35      norm = math.sqrt(x_diff**2+y_diff**2)
36      return x_diff/norm, y_diff/norm
37
38
39  class Bird(pg.sprite.Sprite):
40      """
41      ゲームキャラクター (こうかとん) に関するクラス
42      """
43      delta = { # 押下キーと移動量の辞書
44          pg.K_UP: (0, -1),
```

```
45     pg.K_DOWN: (0, +1),
46     pg.K_LEFT: (-1, 0),
47     pg.K_RIGHT: (+1, 0),
48 }
49
50 ✓ def __init__(self, num: int, xy: tuple[int, int]):
51     """
52     こうかとん画像Surfaceを生成する
53     引数1 num: こうかとん画像ファイル名の番号
54     引数2 xy: こうかとん画像の位置座標タプル
55     """
56     super().__init__()
57     img0 = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
58     img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
59     self.imgs = {
60         (+1, 0): img, # 右
61         (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
62         (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
63         (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
64         (-1, 0): img0, # 左
65         (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
66         (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
67         (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
68     }
69     self.state = "normal"
70     self.hyper_life = 0
71     self.dire = (+1, 0)
72     self.image = self.imgs[self.dire]
73     self.rect = self.image.get_rect()
74     self.rect.center = xy
75     self.speed = 10
76
77
78
79 ✓ def change_img(self, num: int, screen: pg.Surface):
80     """
81     こうかとん画像を切り替え、画面に転送する
82     引数1 num: こうかとん画像ファイル名の番号
83     引数2 screen: 画面Surface
84     """
85     self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
86     screen.blit(self.image, self.rect)
87
88 ✓ def update(self, key_lst: list[bool], screen: pg.Surface):
89     """
90     押下キーに応じてこうかとんを移動させる
91     引数1 key_lst: 押下キーの真理値リスト
92     引数2 screen: 画面Surface
93     """
94     sum_mv = [0, 0]
95
96     for k, mv in __class__.delta.items():
97         if key_lst[k]:
98             sum_mv[0] += mv[0]
99             sum_mv[1] += mv[1]
100         if key_lst[pg.K_LSHIFT]:
101             self.speed = 20
102         else:
103             self.speed = 10
104
105
```

```

105
106         self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
107
108         if check_bound(self.rect) != (True, True):
109             self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
110         if not (sum_mv[0] == 0 and sum_mv[1] == 0):
111             self.dire = tuple(sum_mv)
112             self.image = self.imgs[self.dire]
113         if self.state == "hyper":
114             self.image = pg.transform.laplacian(self.image)
115             self.hyper_life -= 1
116         if self.hyper_life < 0:
117             self.state = "normal"
118         screen.blit(self.image, self.rect)
119
120
121
122     class Bomb(pg.sprite.Sprite):
123         """
124         爆弾に関するクラス
125         """
126         colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
127
128     def __init__(self, emy: "Enemy", bird: Bird):
129         """
130         爆弾円Surfaceを生成する
131         引数1 emy: 爆弾を投下する敵機
132         引数2 bird: 攻撃対象のこうかとん
133         """
134         super().__init__()
135         rad = random.randint(10, 50) # 爆弾円の半径: 10以上50以下の乱数
136         self.image = pg.Surface((2*rad, 2*rad))
137         self.color = random.choice(__class__.colors) # 爆弾円の色: クラス変数からランダム選択
138         pg.draw.circle(self.image, self.color, (rad, rad), rad)
139         self.image.set_colorkey((0, 0, 0))
140         self.rect = self.image.get_rect()
141         # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
142         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
143         self.rect.centerx = emy.rect.centerx
144         self.rect.centery = emy.rect.centery+emy.rect.height/2
145         self.speed = 6
146         self.hp = 1 # HPの追加
147
148     def update(self):
149         """
150         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
151         引数 screen: 画面Surface
152         """
153         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
154         if check_bound(self.rect) != (True, True):
155             self.kill()
156
157
158     class Beam(pg.sprite.Sprite):
159         """
160         ビームに関するクラス
161         """
162     def __init__(self, bird: Bird):
163         """
164         ビーム画像Surfaceを生成する
165         引数 bird: ビームを放つこうかとん。

```

```

165     """
166
167     super().__init__()
168     self.vx, self.vy = bird.dire
169     angle = math.degrees(math.atan2(-self.vy, self.vx))
170     self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 2.0)
171     self.vx = math.cos(math.radians(angle))
172     self.vy = -math.sin(math.radians(angle))
173     self.rect = self.image.get_rect()
174     self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
175     self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
176     self.speed = 10
177
178     def update(self):
179         """
180         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
181         引数 screen : 画面Surface
182         """
183         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
184         if check_bound(self.rect) != (True, True):
185             self.kill()
186
187
188     class Explosion(pg.sprite.Sprite):
189         """
190         爆発に関するクラス
191         """
192         def __init__(self, obj: "Bomb|Enemy", life: int):
193             """
194             爆弾が爆発するエフェクトを生成する
195             引数1 obj : 爆発するBombまたは敵機インスタンス
196             引数2 life : 爆発時間
197             """
198             super().__init__()
199             img = pg.image.load(f"fig/explosion.gif")
200             self.imgs = [img, pg.transform.flip(img, 1, 1)]
201             self.image = self.imgs[0]
202             self.rect = self.image.get_rect(center=obj.rect.center)
203             self.life = life
204
205         def update(self):
206             """
207             爆発時間を1減算した爆発経過時間_lifeに応じて爆発画像を切り替えることで
208             爆発エフェクトを表現する
209             """
210             self.life -= 1
211             self.image = self.imgs[self.life//10%2]
212             if self.life < 0:
213                 self.kill()
214
215
216     class Enemy(pg.sprite.Sprite):
217         """
218         敵機に関するクラス
219         """
220         imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
221
222         def __init__(self):
223             super().__init__()
224             self.image = random.choice(__class__.imgs)
225             self.rect = self.image.get_rect()
```

```

225 self.rect = self.image.get_rect()
226 self.rect.center = random.randint(0, WIDTH), 0
227 self.vy = +6
228 self.bound = random.randint(50, int(HEIGHT/2)) # 停止位置
229 self.state = "down" # 降下状態or停止状態
230 self.interval = random.randint(50, 300) # 爆弾投下インターバル
231 self.hp = 1 # HPの追加
232
233 def update(self):
234     """
235     敵機を速度ベクトルself.vyに基づき移動（降下）させる
236     ランダムに決めた停止位置_boundまで降下したら、_stateを停止状態に変更する
237     引数 screen : 画面Surface
238     """
239     if self.rect.centery > self.bound:
240         self.vy = 0
241         self.state = "stop"
242     self.rect.centery += self.vy
243
244
245 class Score:
246     """
247     打ち落とした爆弾、敵機の数スコアとして表示するクラス
248     爆弾 : 1点
249     敵機 : 10点
250     """
251     def __init__(self):
252         self.font = pg.font.Font(None, 50)
253         self.color = (0, 0, 255)
254         self.value = 100
255         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
256         self.rect = self.image.get_rect()
257         self.rect.center = 100, HEIGHT-50
258
259     def update(self, screen: pg.Surface):
260         self.image = self.font.render(f"Score: {self.value}", 0, self.color)
261         screen.blit(self.image, self.rect)
262
263
264 class Shield(pg.sprite.Sprite):
265     """
266     SPを3消費してこうかとんを守る防御壁を出現させるクラス
267     Caps lock押下で出現
268     """
269
270     def __init__(self, bird : Bird, life):
271         super().__init__()
272         self.life = life
273         self.image = pg.Surface((20, bird.rect.height*2))
274         pg.draw.rect(self.image, (0, 0, 255), (0, 0, 20, bird.rect.height*2))
275
276         vx, vy = bird.dire
277         deg = math.degrees(math.atan2(-vy, vx))
278         self.image = pg.transform.rotozoom(self.image, deg, 1.0)
279         self.image.set_colorkey((0, 0, 0))
280         self.rect = self.image.get_rect()
281         self.rect.centerx = bird.rect.centerx + bird.rect.width * vx
282         self.rect.centery = bird.rect.centery + bird.rect.height * vy
283
284
285     def update(self):

```

```
286         self.life -= 1
287         if self.life < 0:
288             self.kill()
289
290
291 class Gravity(pg.sprite.Sprite):
292     """
293     画面全体を覆う重力場を発生させる
294     """
295     def __init__(self, life):
296         super().__init__()
297         self.image = pg.Surface((1600,900))
298         pg.draw.rect(self.image, (0, 0, 0), (0, 0, 1600,900))
299         self.image.set_alpha(200)
300         self.rect = self.image.get_rect()
301         self.rect.center = (WIDTH/2, HEIGHT/2)
302         self.life = life # 発動時間
303
304     def update(self):
305         self.life -= 1
306         if self.life < 0:
307             self.kill()
308
309
310 class EMP(pg.sprite.Sprite):
311     def __init__(self, Enemy, Bomb, Surface): #敵機、爆弾、surfaceを与えている
312         for emy in Enemy:
313             emy.interval = math.inf
314             emy.image = pg.transform.laplacian(emy.image)
315             emy.image.set_colorkey((0, 0, 0))
316
317         for bomb in Bomb:
318             bomb.speed /= 2
319
320     def update(self):
321         self.life -= 1
322         if self.life < 0:
323             self.kill()
324
325
326 class Powerup:
327     """
328     攻撃力の概念の追加
329     打ち落とした爆弾が赤色だと攻撃力+1
330     """
331     def __init__(self):
332         self.font = pg.font.Font(None, 50)
333         self.color = (0, 0, 255)
334         self.value = 1
335         self.image = self.font.render(f"Power: {self.value}", 0, self.color) # 画面に攻撃力の数値を追加
336         self.rect = self.image.get_rect()
337         self.rect.center = 87, HEIGHT-100
338
339     def update(self, screen: pg.Surface):
340         self.image = self.font.render(f"Power: {self.value}", 0, self.color) # 数値を更新
341         screen.blit(self.image, self.rect)
342
343
344 class Skillpoint:
345     """
```



```
346     スキルポイントの概念の追加
347     打ち落とした爆弾が青色だとSP+1
348     """
349     def __init__(self):
350         self.font = pg.font.Font(None, 50)
351         self.color = (0, 0, 255)
352         self.value = 1
353         self.image = self.font.render(f"SP: {self.value}", 0, self.color) # 画面にスキルポイントの数値を追加
354         self.rect = self.image.get_rect()
355         self.rect.center = 58, HEIGHT-150
356
357     def update(self, screen: pg.Surface):
358         self.image = self.font.render(f"SP: {self.value}", 0, self.color) # 数値を追加
359         screen.blit(self.image, self.rect)
360
361
362     def main():
363         pg.display.set_caption("真！ どうかとん無双")
364         screen = pg.display.set_mode((WIDTH, HEIGHT))
365         bg_img = pg.image.load(f"fig/pg_bg.jpg")
366         score = Score()
367         score.value = 99999 # 実行確認のために仮置き、後で消す
368         power = Powerup()
369         sp = Skillpoint()
370         sp.value = 99999 # 実行確認のために仮置き、後で消す
371
372
373         bird = Bird(3, (900, 400))
374         bombs = pg.sprite.Group()
375         beams = pg.sprite.Group()
376         exps = pg.sprite.Group()
377         emys = pg.sprite.Group()
378         shields = pg.sprite.Group()
379         gravity = pg.sprite.Group()
380
381         tmr = 0
382         clock = pg.time.Clock()
383         while True:
384             key_lst = pg.key.get_pressed()
385             for event in pg.event.get():
386                 if event.type == pg.QUIT:
387                     return 0
388                 if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
389                     beams.add(Beam(bird))
390                 if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT and (sp.value >= 5): # 右シフトキーを
391                     bird.hyper_life = 500
392                     sp.value -= 5 # 消費SP
393                     bird.state = "hyper"
394                 if event.type == pg.KEYDOWN and event.key == pg.K_RETURN and sp.value >= 10: # エンター押したと
395                     gravity.add(Gravity(400))
396                     sp.value -= 10 # 消費SP
397                 if event.type == pg.KEYDOWN and event.key == pg.K_e:
398                     if sp.value > 8:
399                         EMP(emys, bombs, screen)
400                         sp.value -= 8 # 消費SP
401                 if event.type == pg.KEYDOWN and event.key == pg.K_w and sp.value >= 3 and len(shields) == 0: #
402                     sp.value -= 3 # 消費SP
403                     shields.add(Shield(bird, 400))
404                     print(len(shields))
405             screen.blit(bg_img, [0, 0])
```

```
406
407     if tmr%200 == 0: # 200フレームに1回、敵機を出現させる
408         emys.add(Enemy())
409
410     for emy in emys:
411         if emy.state == "stop" and tmr%emy.interval == 0:
412             # 敵機が停止状態に入ったら、intervalに応じて爆弾投下
413             bombs.add(Bomb(emy, bird))
414
415     for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
416         emy.hp -= power.value # 敵のHPを自分の攻撃力分だけ削る
417         if emy.hp <= 0: # 敵のHPが0以下の時
418             exps.add(Explosion(emy, 100)) # 爆発エフェクト
419             score.value += 10 # 10点アップ
420             bird.change_img(6, screen) # こうかたん喜びエフェクト
421
422     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
423         bomb.hp -= power.value # 爆弾の耐久力を自分の攻撃力分だけ削る
424         if bomb.hp <= 0: # 爆弾の耐久力が0以下の時
425             exps.add(Explosion(bomb, 50)) # 爆発エフェクト
426             score.value += 1 # 1点アップ
427         if bomb.color == (255, 0, 0): # 敵の爆弾の色が赤色のとき
428             power.value += 1 # 攻撃力アップ
429         if bomb.color == (0, 0, 255): # 敵の爆弾の色が青色のとき
430             sp.value += 1 # スキルポイントアップ
431         """
432         HPのクラスが追加されたら追加する
433         今回はマージできないので追加しない
434         """
435         #if bomb.color == (0, 255, 0):
436         #    hp.value += 1 # HP回復
437
438     for bomb in pg.sprite.spritecollide(bird, bombs, True):
439         if bird.state == "hyper":
440             exps.add(Explosion(bomb, 50))
441             score.value += 1
442         if bird.state == "normal":
443             bird.change_img(8, screen) # こうかたん悲しみエフェクト
444             score.update(screen)
445             pg.display.update()
446             time.sleep(2)
447             return
448     for bomb in pg.sprite.groupcollide(bombs, gravity, True, False).keys():
449         bomb.hp -= power.value # 爆弾の耐久力を自分の攻撃力分だけ削る
450         if bomb.hp <= 0: # 爆弾の耐久力が0以下の時
451             exps.add(Explosion(bomb, 50))
452             score.value += 1
453
454     for emy in pg.sprite.groupcollide(emys, gravity, True, False).keys():
455         emy.hp -= power.value # 敵のHPを自分の攻撃力分だけ削る
456         if emy.hp <= 0: # 敵のHPが0以下の時
457             exps.add(Explosion(emy, 100))
458             score.value += 10
459             bird.change_img(6, screen) # こうかたん喜びエフェクト
460
461
462     for bomb in pg.sprite.groupcollide(bombs, shields, True, False).keys():
463         exps.add(Explosion(bomb, 50))
464         score.value += 1
465
```

```
466         if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
467             bird.change_img(8, screen) # こうかたん悲しみエフェクト
468             score.update(screen)
469             pg.display.update()
470             time.sleep(2)
471             return
472
473
474         bird.update(key_lst, screen)
475         beams.update()
476         beams.draw(screen)
477         emys.update()
478         emys.draw(screen)
479         bombs.update()
480         bombs.draw(screen)
481         exps.update()
482         exps.draw(screen)
483         gravity.update()
484         gravity.draw(screen)
485         score.update(screen)
486         shields.update()
487         shields.draw(screen)
488         power.update(screen)
489         sp.update(screen)
490
491         pg.display.update()
492         tmr += 1
493         clock.tick(50)
494
495
496     if __name__ == "__main__":
497         pg.init()
498         main()
499         pg.quit()
500         sys.exit()
```